



# UTM

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING  
SEMESTER II, SESSION 2024/2025

BACHELOR OF COMPUTER SCIENCE (BIOINFORMATICS)

SECB3203 PROGRAMMING FOR BIOINFORMATICS - SECTION 01

**PROJECT  
ALZHEIMER'S DISEASE PREDICTION**

GROUP MEMBERS	MATRIC NO
TAN ZHAO HONG	A23CS0188
CHIN PEI WEN	A23CS0065
KOO XUAN	A23CS0300

LECTURER'S NAME : DR. SEAH CHOON SEN

SUBMISSION DATE : 14 JANUARY 2025

## **Table of Contents**

<b>1.0 Introduction.....</b>	<b>3</b>
<b>1.1 Problem Background.....</b>	<b>3</b>
<b>1.2 Problem Statement.....</b>	<b>4</b>
<b>1.3 Objectives.....</b>	<b>5</b>
<b>1.4 Scopes.....</b>	<b>5</b>
<b>2.0 Data Collection and Pre-processing.....</b>	<b>8</b>
<b>2.1 Importing Dataset.....</b>	<b>9</b>
<b>2.2 Data wrangling.....</b>	<b>13</b>
<b>2.3 Software and Hardware Requirements.....</b>	<b>21</b>
<b>3.0 Flowchart of the Proposed Approach.....</b>	<b>23</b>
<b>3.1 Exploratory Data Analysis.....</b>	<b>24</b>
3.1.1 Descriptive Statistics.....	24
3.1.2 Basic of Grouping.....	24
3.1.3 ANOVA.....	26
3.1.4 Correlation.....	26
<b>3.2 Model Development.....</b>	<b>28</b>
3.2.1 Imported Libraries.....	28
3.2.2 Data Preparation for Model Development.....	29
3.2.3 Classification models implemented for Alzheimer's disease prediction.....	30
<b>3.3 Model Evaluation.....</b>	<b>32</b>
3.3.1 Evaluation Metrics.....	32
3.3.2 Confusion Matrix.....	34
3.3.3 Model Performance Comparison.....	35
3.3.4 ROC Curve Analysis.....	36
3.3.5 Prediction and Decision Making.....	37
<b>4.0 Testing and Validation.....</b>	<b>38</b>
<b>5.0 Conclusion.....</b>	<b>39</b>
<b>6.0 Reference.....</b>	<b>41</b>

# 1.0 Introduction

Alzheimer's Disease (AD) represents one of the most pressing challenges in modern healthcare, being the most prevalent form of dementia worldwide. As the global population ages, the incidence of AD continues to rise dramatically, affecting millions of individuals and their families. Early and accurate diagnosis of Alzheimer's Disease is critical for timely intervention, treatment planning, and improving patient quality of life.

This project aims to develop a robust machine learning-based classification system for Alzheimer's Disease prediction using comprehensive clinical, demographic, and lifestyle data. By leveraging advanced computational techniques and bioinformatics approaches, we seek to create an effective predictive model that can assist healthcare professionals in early AD detection and risk assessment.

The availability of large-scale healthcare datasets and advancements in machine learning algorithms present unprecedented opportunities to identify patterns and biomarkers associated with Alzheimer's Disease. Through this project, we will explore various machine learning techniques, including feature selection, data preprocessing, model development, and evaluation to create an accurate and interpretable classification system.

## 1.1 Problem Background

Alzheimer's Disease is a progressive neurodegenerative disorder characterized by cognitive decline, memory loss, and behavioral changes. According to recent studies, AD affects approximately 35.6% of diagnosed dementia cases, with the remaining 64.4% representing individuals without the disease. The challenge lies in the complexity of AD diagnosis, which traditionally relies on:

- **Clinical assessments:** Cognitive and functional tests that may be subjective
- **Medical imaging:** MRI and PET scans that require specialized equipment and expertise
- **Medical history:** Comprehensive patient background that may be incomplete
- **Laboratory tests:** Biomarker analysis that can be expensive and time-consuming

Traditional diagnostic methods often result in:

1. **Late-stage diagnosis:** Many patients are diagnosed only after significant cognitive decline
2. **Inconsistent results:** Variability between different assessment methods and practitioners
3. **Resource constraints:** Limited access to specialized diagnostic facilities, particularly in rural or underserved areas
4. **High costs:** Expensive imaging and laboratory procedures that may not be covered by insurance

The dataset we will utilize contains records from 2,149 patients with 34 comprehensive features spanning multiple domains:

- **Demographic factors:** Age, gender, ethnicity, education level
- **Lifestyle factors:** Physical activity, diet quality, sleep patterns, alcohol consumption, smoking status
- **Medical history:** Family history of AD, cardiovascular disease, diabetes, hypertension, depression
- **Clinical measurements:** BMI, blood pressure, cholesterol levels
- **Cognitive assessments:** MMSE scores, functional assessment scores, memory complaints
- **Symptoms:** Behavioral changes, confusion, disorientation, personality changes

This rich dataset provides an opportunity to develop machine learning models that can identify patterns and risk factors associated with Alzheimer's Disease, potentially enabling earlier detection and intervention.

## 1.2 Problem Statement

Current Alzheimer's Disease diagnostic approaches face several critical challenges:

1. **Delayed diagnosis:** Most patients are diagnosed in moderate to advanced stages when cognitive decline is already significant, limiting the effectiveness of available interventions
2. **Limited accessibility:** Advanced diagnostic tools such as specialized imaging (PET scans, advanced MRI) and biomarker testing are not readily available in all healthcare settings, particularly in developing regions
3. **High diagnostic costs:** Comprehensive AD assessment can be prohibitively expensive, involving multiple specialist consultations, imaging studies, and laboratory tests
4. **Subjective assessments:** Many cognitive tests rely on clinical judgment, which can vary between practitioners and may be influenced by patient demographics and socioeconomic factors
5. **Data integration challenges:** Patient information is often scattered across multiple sources (clinical notes, lab results, imaging reports), making it difficult to obtain a holistic view for accurate diagnosis
6. **Lack of predictive tools:** Current methods primarily focus on confirming diagnosis rather than predicting risk, missing opportunities for preventive interventions

## **1.3 Objectives**

The primary purpose of this project is to explore how computational and bioinformatics approaches can be applied to analyse clinical datasets and support biomedical research. The project aims :

- To demonstrate practical use of programming tools for data cleaning, data interpretation, and predictive modelling.
- To provide insights into how machine-learning methods may assist in distinguishing between different stages of Alzheimer's disease.
- To cultivate understanding of how computational models contribute to addressing real healthcare challenges.

## **1.4 Scopes**

This project focuses on the application of programming and machine learning techniques in the field of bioinformatics to predict Alzheimer's disease using clinical data. The scope of the study is defined as follows:

### **Domain of Data**

The domain of this study is healthcare and bioinformatics, specifically within the area of neurodegenerative disease analysis. The dataset contains clinical, demographic, lifestyle, and cognitive assessment data related to Alzheimer's disease, enabling computational analysis for medical decision support.

### **Time Frame of Data**

The dataset used in this project is obtained from a publicly available Kaggle repository and represents historical patient records collected over multiple years. The study does not involve real-time or longitudinal tracking of patients, and all analyses are conducted on static, retrospective data.

### **Type of Data Considered**

This study utilizes structured tabular data stored in spreadsheet format (.xlsx, converted to .csv). The data consists of both numerical and categorical variables, including continuous measurements, binary indicators, and ordinal attributes. No unstructured data such as medical images, text records, or genetic sequences are included.

### **Attributes Used**

The attributes used in this study are limited to medically relevant features, including:

- Demographic information (e.g., age, gender, education level)
- Lifestyle factors (e.g., physical activity, diet quality, sleep patterns)

- Medical history (e.g., family history of Alzheimer's disease, cardiovascular disease, diabetes)
- Clinical measurements (e.g., BMI, blood pressure, cholesterol levels)
- Cognitive and functional assessments (e.g., MMSE score, functional assessment score)
- Non-medical identifiers and administrative attributes were excluded to reduce noise and potential bias.

## **Area of Study**

The study lies within the intersection of computer science, bioinformatics, and medical data analysis, with an emphasis on disease prediction and clinical decision support systems. It demonstrates how computational methods can assist in early detection of Alzheimer's disease.

## **Techniques Used**

The project applies several data science and machine learning techniques, including:

- Data preprocessing (missing value handling, normalization, encoding)
- Exploratory Data Analysis (descriptive statistics, grouping, correlation analysis, ANOVA)
- Supervised machine learning classification
- Performance evaluation using statistical and graphical methods

## **Methodology and Approach**

A systematic machine learning pipeline is adopted in this study. The methodology includes data collection, preprocessing, exploratory analysis, feature transformation, model training, evaluation, and comparison. A comparative approach is used to assess multiple classification algorithms to determine the most effective model for Alzheimer's disease prediction.

## **Testing, Measurement, and Experimental Tools**

Model testing and evaluation are performed using:

- Train-test split with an 80:20 ratio
- Cross-validation for performance stability
- Evaluation metrics such as accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC
- The experiments are conducted using Python libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn.

## **Test Data**

The test data consists of 20% of the original dataset, which is unseen during model training. Stratified sampling is applied to preserve the class distribution of Alzheimer's and non-Alzheimer's cases, ensuring fair and reliable evaluation.

## **Limits of the Research**

This study has several limitations:

- The dataset is limited to a single publicly available source and may not represent all populations.
- The analysis relies on structured clinical data and does not include imaging, genetic, or biomarker data.
- The models are designed for prediction and risk assessment rather than clinical diagnosis.
- External validation using independent datasets is not performed.
- The results are dependent on data quality and feature availability within the dataset. Gain hands-on experience working with health-related datasets.

## 2.0 Data Collection and Pre-processing

Our dataset was fetched from Kaggle:

<https://www.kaggle.com/datasets/rabieelkharoua/alzheimers-disease-dataset/data>

The dataset is in .xlsx, thus we convert it to csv file so that it can be easily read by Python.

## 2.1 Importing Dataset

- Python packages for Data Science

```
4 import numpy as np
5 import pandas as pd
6 import matplotlib.pyplot as plt
7 import seaborn as sns
8 from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
9 from sklearn.preprocessing import StandardScaler, LabelEncoder
10 from sklearn.metrics import (accuracy_score, precision_score, recall_score,
11 | | | | | | | | | | | | f1_score, confusion_matrix, classification_report,
12 | | | | | | | | | | | | roc_auc_score, roc_curve)
13 # from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
14 from sklearn.linear_model import LogisticRegression
15 from sklearn.svm import SVC
16 from sklearn.tree import DecisionTreeClassifier
17 from sklearn.neighbors import KNeighborsClassifier
18 import warnings
19 warnings.filterwarnings('ignore')
```

- pandas: Data manipulation and analysis
- numpy: Numerical operations and array handling
- matplotlib: Data visualization
- seaborn: Statistical data visualization
- scikit-learn: Machine learning algorithms and preprocessing

## - Importing Dataset File

```
37  # =====
38  # 1. DATA LOADING
39  # =====
40
41 ✓ def load_data(filepath):
42     """Load the Alzheimer's disease dataset"""
43     df = pd.read_csv(filepath)
44     print(f"Dataset shape: {df.shape}")
45     print(df)
46
| 367 ✓ if __name__ == "__main__":
368     FILEPATH = "alzheimers_disease_data.csv"
369     TARGET_COLUMN = 'Diagnosis'
```

```
=====
ALZHEIMER'S DISEASE PREDICTION - ML PIPELINE
=====

[1] Loading Data...
Dataset shape: (2149, 35)
   PatientID  Age  Gender  Ethnicity  EducationLevel    BMI ... Disorientation  PersonalityChanges  DifficultyCompletingTasks  Forgetfulness  Diagnosis  DoctorInCharge
0      4751  73       0        0           2  22.927749 ...          0            0             1            0            0            0            0            XXXConfid
1      4752  89       0        0           0  26.827681 ...          0            0             0            1            0            0            0            XXXConfid
2      4753  73       0        3           1  17.795882 ...          1            0             1            0            0            0            0            XXXConfid
3      4754  74       1        0           1  33.800817 ...          0            0             0            0            0            0            0            XXXConfid
4      4755  89       0        0           0  20.716974 ...          0            0             1            0            0            0            0            XXXConfid
...
...  ...
...  ...
...  ...
...  ...
2144  6895  61       0        0           1  39.121757 ...          0            0             0            0            1            0            1            XXXConfid
2145  6896  75       0        0           2  17.857903 ...          0            0             0            0            0            1            1            XXXConfid
2146  6897  77       0        0           1  15.476479 ...          0            0             0            0            0            0            1            XXXConfid
2147  6898  78       1        3           1  15.299911 ...          0            0             0            0            1            1            1            XXXConfid
2148  6899  72       0        0           2  33.289738 ...          1            0             0            0            0            1            0            XXXConfid
[2149 rows x 35 columns]
```

These screenshots show the initial dataset loading process.

- Dataset Information

```
46 |     print(f"\nDataset info:")
47 |     print(df.info())
```

```
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2149 entries, 0 to 2148
Data columns (total 35 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   PatientID        2149 non-null    int64  
 1   Age              2149 non-null    int64  
 2   Gender            2149 non-null    int64  
 3   Ethnicity         2149 non-null    int64  
 4   EducationLevel   2149 non-null    int64  
 5   BMT              2149 non-null    float64 
 6   Smoking           2149 non-null    int64  
 7   AlcoholConsumption 2149 non-null    float64 
 8   PhysicalActivity  2149 non-null    float64 
 9   DietQuality       2149 non-null    float64 
 10  SleepQuality      2149 non-null    float64 
 11  FamilyHistoryAlzheimers 2149 non-null    int64  
 12  CardiovascularDisease 2149 non-null    int64  
 13  Diabetes          2149 non-null    int64  
 14  Depression         2149 non-null    int64  
 15  HeadInjury         2149 non-null    int64  
 16  Hypertension        2149 non-null    int64  
 17  SystolicBP         2149 non-null    int64  
 18  DiastolicBP        2149 non-null    int64  
 19  CholesterolTotal   2149 non-null    float64 
 20  CholesterolLDL    2149 non-null    float64 

 21  CholesterolHDL    2149 non-null    float64 
 22  CholesterolTriglycerides 2149 non-null    float64 
 23  MMSE              2149 non-null    float64 
 24  FunctionalAssessment 2149 non-null    float64 
 25  MemoryComplaints  2149 non-null    int64  
 26  BehavioralProblems 2149 non-null    int64  
 27  ADL               2149 non-null    float64 
 28  Confusion          2149 non-null    int64  
 29  Disorientation      2149 non-null    int64  
 30  PersonalityChanges 2149 non-null    int64  
 31  DifficultyCompletingTasks 2149 non-null    int64  
 32  Forgetfulness       2149 non-null    int64  
 33  Diagnosis          2149 non-null    int64  
 34  DoctorInCharge     2149 non-null    object  
dtypes: float64(12), int64(22), object(1)
memory usage: 587.7+ KB
None
```

These screenshots show the dataset information details.

- Feature Removal

```
25  # =====
26  # FEATURES TO REMOVE (Non-medical and weak evidence features)
27  # =====
28 FEATURES_TO_REMOVE = [
29     'PatientID',           # Administrative identifier
30     'DoctorInCharge',      # Administrative data
31     'Gender',              # Weak predictor
32     'Ethnicity',           # May introduce bias, weak medical relevance
33     'AlcoholConsumption',   # Inconsistent evidence
34     'SleepQuality'         # Weak/inconsistent evidence
35 ]
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51     # Remove non-medical and weak evidence features
52     print(f"\n{'='*80}")
53     print("REMOVING NON-MEDICAL AND WEAK EVIDENCE FEATURES")
54     print(f"{'='*80}")
55     features_removed = [col for col in FEATURES_TO_REMOVE if col in df.columns]
56     if features_removed:
57         print(f"\nRemoving features: {features_removed}")
58         df = df.drop(columns=features_removed)
59         print(f"New dataset shape: {df.shape}")
60     else:
61         print("\nNo features to remove (features not found in dataset)")
62
63     return df
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
636
```

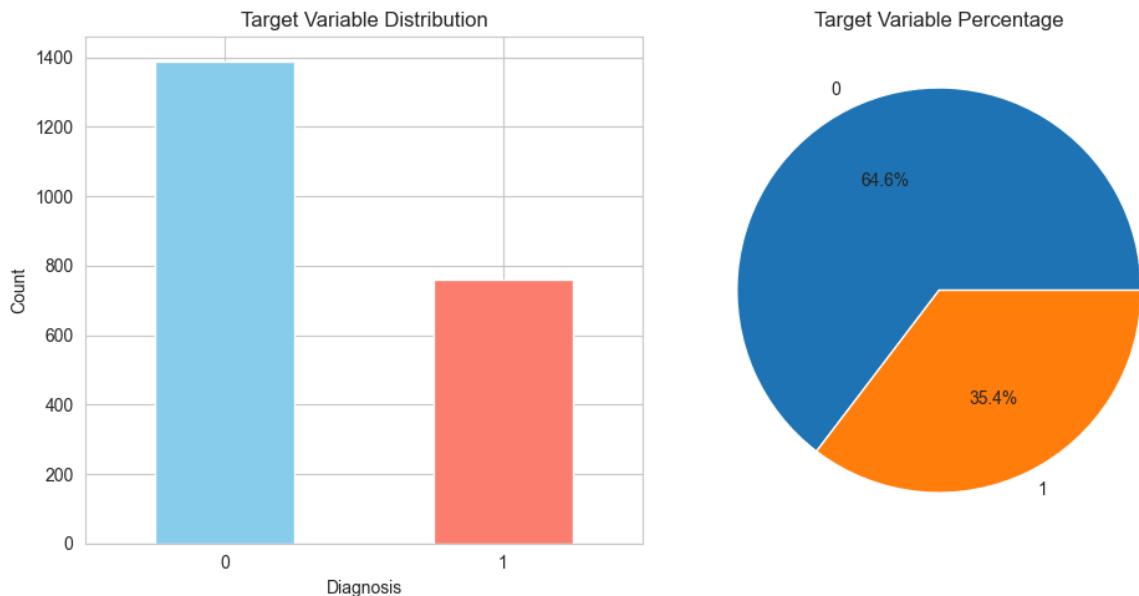
```
=====  
REMOVING NON-MEDICAL AND WEAK EVIDENCE FEATURES  
=====  
  
Removing features: ['PatientID', 'DoctorInCharge', 'Gender', 'Ethnicity', 'AlcoholConsumption', 'SleepQuality']  
New dataset shape: (2149, 29)
```

These screenshots show the removal of non-medical and weak-evidence features such as patient identifiers and administrative attributes. These features were excluded to reduce noise, prevent bias, and ensure that the model focuses only on medically relevant variables that contribute meaningfully to Alzheimer's disease prediction.

## 2.2 Data wrangling

- Target Variable Distribution

```
110     def preprocess_data(df, target_column='Diagnosis'):
111         """Preprocess the data for machine learning"""
112
113         # Separate features and target
114         X = df.drop(columns=[target_column])
115         y = df[target_column]
116
117         # Encode target variable if it's categorical
118         if y.dtype == 'object':
119             print(f"\nEncoding target variable")
120             le_target = LabelEncoder()
121             y = le_target.fit_transform(y)
```



This figure illustrates the distribution of the target variable (Diagnosis) using both bar and pie charts. Visualizing the class distribution helps identify potential class imbalance and confirms that the problem is a binary classification task, which is suitable for the selected machine learning models.

- Identifying and handling missing values

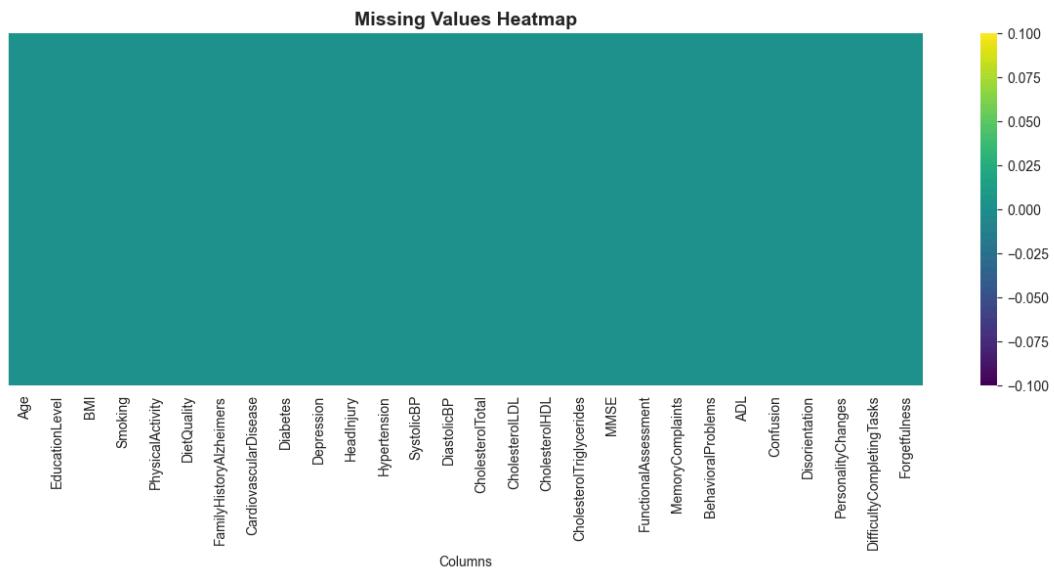
```

123 # VISUALIZE MISSING VALUES
124 print("\n--- Identifying and Handling Missing Values ---")
125 print("\n1. Missing Values Count:")
126 print(x.isnull().sum())
127
128 print("\n2. Visualizing Missing Values:")
129 plt.figure(figsize=(12, 6))
130 sns.heatmap(x.isnull(), cbar=True, cmap='viridis', yticklabels=False)
131 plt.title('Missing Values Heatmap', fontsize=14, fontweight='bold')
132 plt.xlabel('Columns')
133 plt.tight_layout()
134 plt.savefig('missing_values_heatmap.png', dpi=300)
135 plt.show()
136
137 # Handle missing values
138 if x.isnull().sum().sum() > 0:
139     print("\n3. Handling missing values...")
140     x = x.fillna(x.median())
141     print("Missing values after handling:")
142     print(x.isnull().sum())
143 else:
144     print("\n3. No missing values detected!")

```

## 2. Visualizing Missing Values:

3. No missing values detected!



These screenshots display the detection of missing values across all input features. Median imputation was applied to numerical attributes to handle missing data, as it is robust to outliers and preserves the overall distribution of the dataset. This step ensures data completeness before further processing.

- Data Formatting

```
146     # DATA FORMATTING
147     print("\n--- Data Formatting ---")
148     print("\n1. Current Data Types:")
149     print(x.dtypes)
150
151     print("\n2. Numerical Columns:")
152     numerical_cols = x.select_dtypes(include=[np.number]).columns
153     print(list(numerical_cols))
154
155     print("\n3. Categorical Columns:")
156     categorical_cols_orig = x.select_dtypes(include=['object']).columns
157     print(list(categorical_cols_orig))
```

```
--- Data Formatting ---
1. Current Data Types:
Age                      int64
EducationLevel            int64
BMT                      float64
Smoking                  int64
PhysicalActivity          float64
DietQuality              float64
FamilyHistoryAlzheimers   int64
CardiovascularDisease    int64
Diabetes                 int64
Depression               int64
HeadInjury                int64
Hypertension              int64
SystolicBP                int64
DiastolicBP              int64
CholesterolTotal          float64
CholesterolLDL           float64
CholesterolHDL           float64
CholesterolTriglycerides float64
MMSE                     float64
FunctionalAssessment      float64
MemoryComplaints          int64
BehavioralProblems        int64
ADL                      float64
Confusion                 int64
Disorientation             int64
PersonalityChanges         int64
DifficultyCompletingTasks int64
Forgetfulness              int64
dtype: object

2. Numerical Columns:
['Age', 'EducationLevel', 'BMT', 'Smoking', 'PhysicalActivity', 'DietQuality', 'FamilyHistoryAlzheimers', 'cardiovascularDisease', 'Diabetes', 'Depression', 'HeadInjury', 'Hypertension', 'SystolicBP', 'DiastolicBP', 'CholesterolTotal', 'CholesterolLDL', 'CholesterolHDL', 'CholesterolTriglycerides', 'MMSE', 'FunctionalAssessment', 'MemoryComplaints', 'BehavioralProblems', 'ADL', 'Confusion', 'Disorientation', 'PersonalityChanges', 'DifficultyCompletingTasks', 'Forgetfulness']

3. Categorical columns:
[]
```

This step verifies and formats the dataset by inspecting data types and separating numerical and categorical features. Ensuring correct data types is essential before applying normalization, binning, and encoding, as machine learning algorithms require numerical and consistently formatted inputs.

- Data Normalization

```
159     # DATA NORMALIZATION
160     print("\n--- Data Normalization ---")
161
162     # Import MinMaxScaler
163     from sklearn.preprocessing import MinMaxScaler
164
165     print("\n1. Min-Max Scaling (Normalization)")
166     print("  Formula:  $x_{scaled} = (x - x_{min}) / (x_{max} - x_{min})$ ")
167     print("  Result: Values scaled to range [0, 1]")
168
169     if 'Age' in X.columns:
170         # Create a copy for demonstration
171         min_max_scaler = MinMaxScaler()
172         age_minmax = min_max_scaler.fit_transform(X[['Age']])
173
174         print("\n  Example: Age Column")
175         print(f"  Original range: [{X['Age'].min():.2f}, {age_minmax.min():.2f}]")
176         print(f"  Scaled range: [{age_minmax.min():.2f}, {age_minmax.max():.2f}]")
177
178
179     # Show comparison
180     comparison_df = pd.DataFrame({
181         'Original_Age': X['Age'].head(10),
182         'MinMax_Scaled': age_minmax[:10].flatten()
183     })
184     print("\n  Sample comparison:")
185     print(comparison_df)
186
187     # Visualize
188     fig, axes = plt.subplots(1, 2, figsize=(14, 5))
189     axes[0].hist(X['Age'], bins=20, color='skyblue', edgecolor='black')
190     axes[0].set_title('Original Age Distribution', fontsize=12, fontweight='bold')
191     axes[0].set_xlabel('Age')
192     axes[0].set_ylabel('Frequency')
193
194     axes[1].hist(age_minmax, bins=20, color='salmon', edgecolor='black')
195     axes[1].set_title('Min-Max Scaled Age [0,1]', fontsize=12, fontweight='bold')
196     axes[1].set_xlabel('Scaled Age')
197     axes[1].set_ylabel('Frequency')
198
199     plt.tight_layout()
200     plt.savefig('minmax_scaling.png', dpi=300)
201     plt.show()
202
203     print("\n2. Z-Score Normalization (Standardization)")
204     print("  Formula:  $z = (x - \mu) / \sigma$ ")
205     print("  Note: This will be applied later using StandardScaler")
```

### --- Data Normalization ---

#### 1. Min-Max Scaling (Normalization)

Formula:  $x_{scaled} = (x - x_{min}) / (x_{max} - x_{min})$

Result: Values scaled to range [0, 1]

Example: Age column

Original range: [60.00, 90.00]

Scaled range: [0.00, 1.00]

Sample comparison:

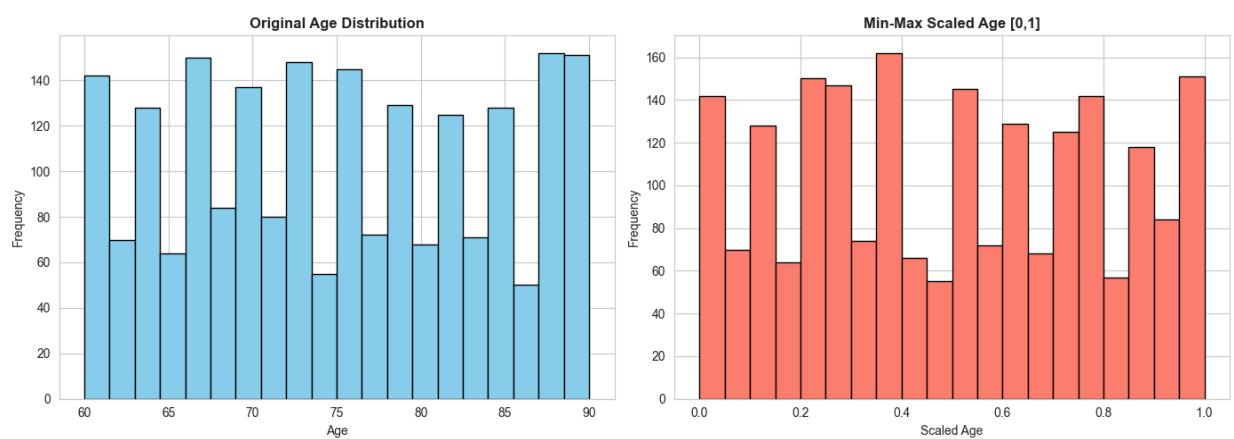
Original\_Age MinMax\_Scaled

0	73	0.433333
1	89	0.966667
2	73	0.433333
3	74	0.466667
4	89	0.966667
5	86	0.866667
6	68	0.266667
7	75	0.500000
8	72	0.400000
9	87	0.900000

#### 2. Z-Score Normalization (Standardization)

Formula:  $z = (x - \mu) / \sigma$

Note: This will be applied later using StandardScaler



This output demonstrates Min-Max normalization applied to the Age feature, scaling values into the range [0,1]. The comparison between original and scaled values highlights how normalization prevents features with larger numerical ranges from dominating the learning process of distance-based and gradient-based models.

- Binning

```
206     # BINNING
207     print("\n--- Binning ---")
208
209     if 'Age' in X.columns:
210         print("\n1. Age Binning:")
211         X['Age_binned'] = pd.cut(
212             X['Age'],
213             bins=[0, 60, 75, 100],
214             labels=['Young', 'Middle', 'Old']
215         )
216
217         # Show binning results
218         print("\n    Sample of Age Binning:")
219         print(X[['Age', 'Age_binned']].head(10))
220
221         print("\n    Age Group Distribution:")
222         print(X['Age_binned'].value_counts())
223
224         # Visualize
225         plt.figure(figsize=(10, 6))
226         X['Age_binned'].value_counts().sort_index().plot(kind='bar', color='steelblue', edgecolor='black')
227         plt.title('Age Group Distribution After Binning', fontsize=14, fontweight='bold')
228         plt.xlabel('Age Group')
229         plt.ylabel('Count')
230         plt.xticks(rotation=0)
231         plt.tight_layout()
232         plt.savefig('age_binning.png', dpi=300)
233         plt.show()
```

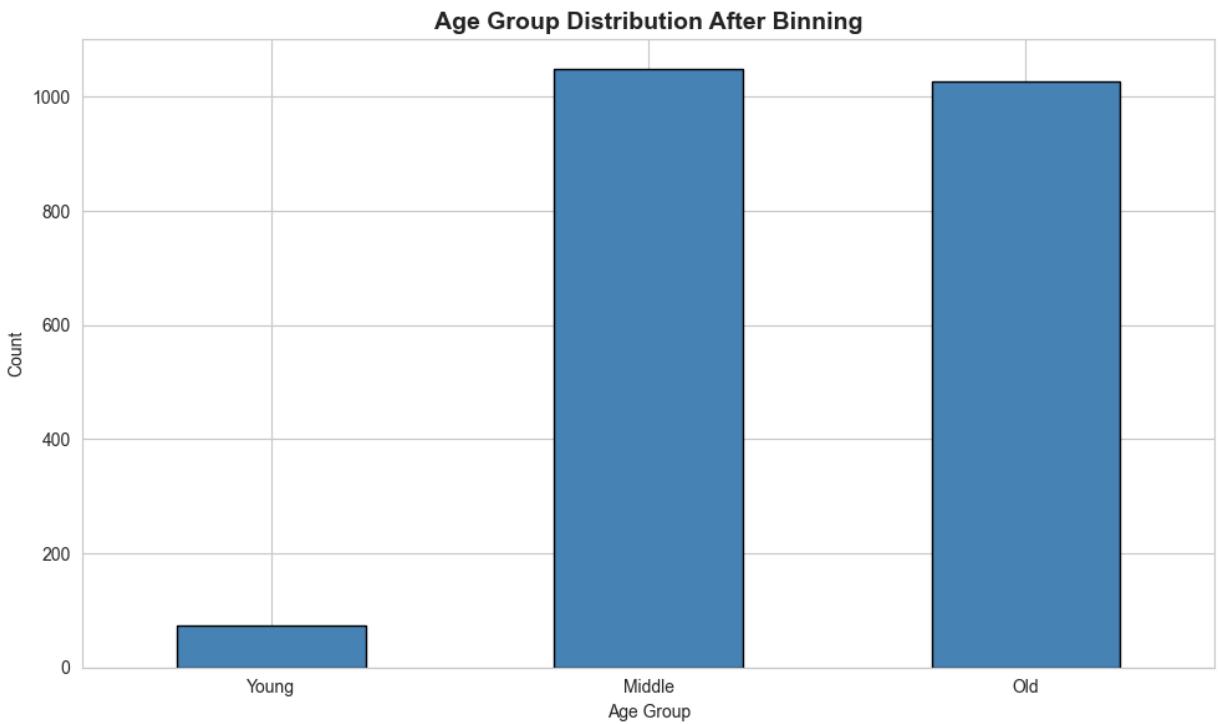
```
--- Binning ---
```

```
1. Age Binning:
```

```
    Sample of Age Binning:
    Age  Age_binned
0    73      Middle
1    89        Old
2    73      Middle
3    74      Middle
4    89        Old
5    86        Old
6    68      Middle
7    75      Middle
8    72      Middle
9    87        Old
```

```
    Age Group Distribution:
```

```
    Age_binned
    Middle    1048
    Old      1027
    Young      74
Name: count, dtype: int64
```



In this step, the continuous Age feature was transformed into categorical age groups using binning. This approach helps capture non-linear patterns related to age and Alzheimer's risk, improves interpretability, and allows the model to learn differences between broader age groups rather than relying only on exact numerical values.

- Indicator variables

```
235 # INDICATOR VARIABLES
236 print("\n--- Indicator Variables (One-Hot Encoding) ---")
237
238 categorical_cols = X.select_dtypes(include=['object', 'category']).columns
239
240 if len(categorical_cols) > 0:
241     print(f"\n1. Categorical columns to encode: {list(categorical_cols)}")
242
243     # Show before encoding
244     print("\n2. Before encoding (first 5 rows):")
245     print(X[categorical_cols].head())
246
247     # Apply one-hot encoding
248     X = pd.get_dummies(X, columns=categorical_cols, drop_first=True)
249
250     print(f"\n3. After one-hot encoding:")
251     print(f"    Original features: {len(categorical_cols)}")
252     print(f"    New indicator columns created: {len([col for col in X.columns if any(cat in col for cat in categorical_cols)])}")
253     print(f"    Total columns now: {X.shape[1]}")
```

--- Indicator Variables (One-Hot Encoding) ---

1. Categorical columns to encode: ['Age\_binned']

2. Before encoding (first 5 rows):

Age\_binned

0	Middle
1	Old
2	Middle
3	Middle
4	Old

3. After one-hot encoding:

Original features: 1

New indicator columns created: 2

Total columns now: 30

These screenshots show the conversion of categorical features created through binning into numerical indicator variables using one-hot encoding. This transformation is necessary because machine learning models require numerical inputs and cannot directly process categorical text labels.

- Train–Test Split and Z-Score Normalization

```
254  
255     # SPLIT AND SCALE  
256     # Split the data  
257     X_train, X_test, y_train, y_test = train_test_split(  
258         X, y, test_size=0.2, random_state=42, stratify=y  
259     )  
260  
261     # Scale the features (Z-score normalization)  
262     print("\n--- Final Scaling (Z-Score Normalization) ---")  
263     scaler = StandardScaler()  
264     X_train_scaled = scaler.fit_transform(X_train)  
265     X_test_scaled = scaler.transform(X_test)  
266  
267     print(f"\nTraining set size: {X_train_scaled.shape}")  
268     print(f"Test set size: {X_test_scaled.shape}")  
269  
270     print("\n" + "="*80)  
271     print("DATA PREPROCESSING COMPLETE")  
272     print("="*80)  
273  
274     return X_train_scaled, X_test_scaled, y_train, y_test, X.columns  
275
```

```
--- Final Scaling (Z-Score Normalization) ---
```

```
Training set size: (1719, 30)  
Test set size: (430, 30)
```

```
=====
```

```
DATA PREPROCESSING COMPLETE
```

```
=====
```

The dataset was split into training and testing sets using stratified sampling to preserve class distribution. Z-score normalization was then applied using StandardScaler to standardize features, ensuring that all variables contribute equally during model training.

## **2.3 Software and Hardware Requirements**

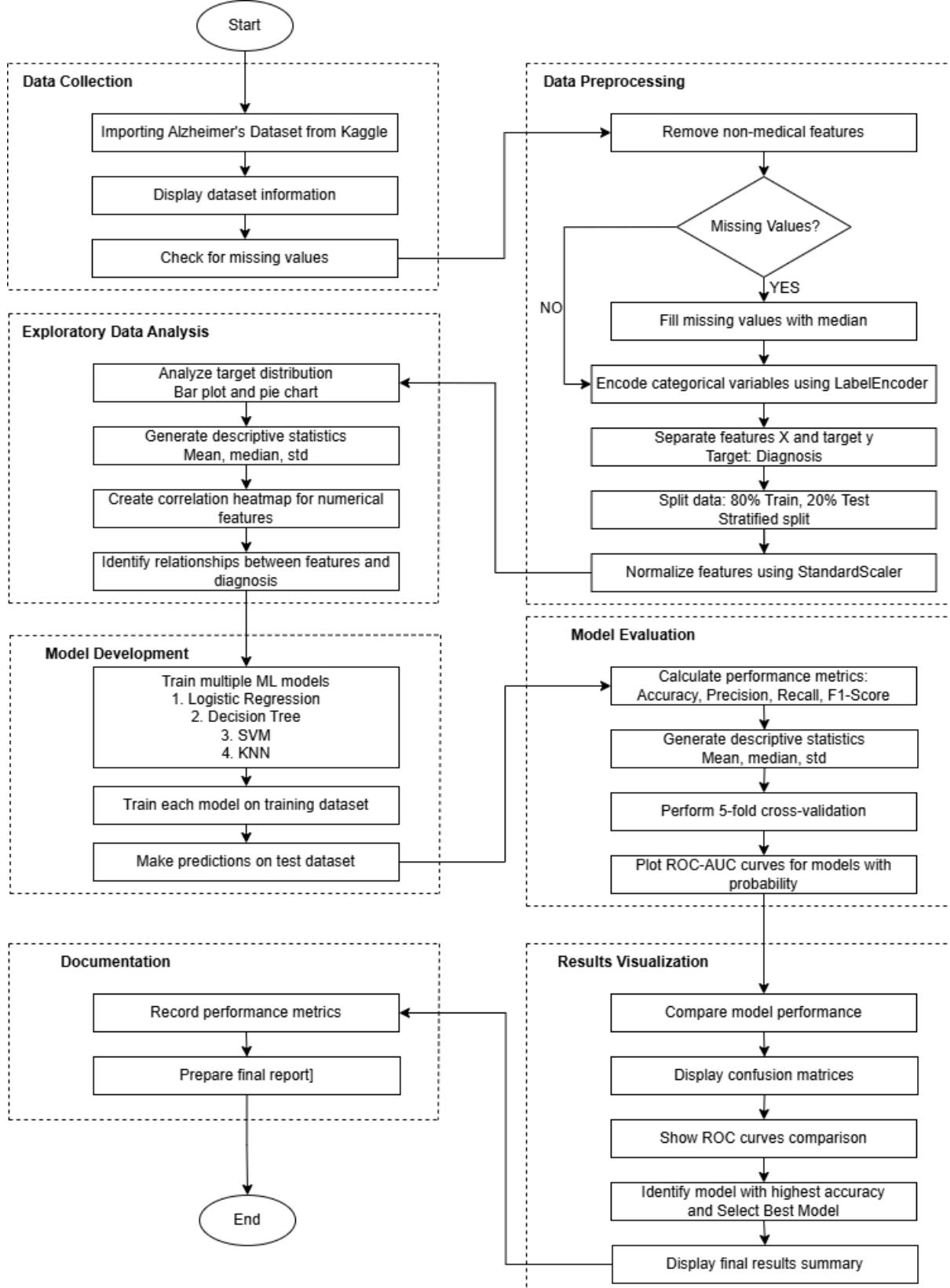
### **2.3.1 Software Tools**

- Visual Studio Code
  - Integrated Development Environment (IDE) for script development and project organization
- Python
  - Programming language used for data preprocessing, analysis, and machine learning model development.
- Jupyter Notebook
  - Environment for code execution, data visualization, and reproducibility of experiments.
- Python libraries
  - Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn for data manipulation, visualization, and machine learning tasks

### **2.3.2 Hardware Specifications**

- Computer
  - System type: 64-bit operating system
  - Processor: 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz
  - Memory (RAM): 8GB, sufficient for handling datasets and training classical machine learning models
  - Storage: 476GB, adequate for storing datasets, model parameters, and project files

### 3.0 Flowchart of the Proposed Approach



### 3.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an essential step in understanding the structure, characteristics, and patterns within the dataset before building machine learning models. In this project, EDA is conducted using Pandas, NumPy, Matplotlib, and Seaborn to explore the Alzheimer's Disease dataset. The analysis focuses on descriptive statistics, data grouping, analysis of variance (ANOVA), and correlation analysis.

#### 3.1.1 Descriptive Statistics

Descriptive statistics are used to summarize the main characteristics of the dataset numerically. Using the `df.describe()` function from Pandas, statistical measures such as mean, standard deviation, minimum, maximum, and quartiles are calculated for all numerical attributes.

```
101      # Statistical summary
102      print("\nStatistical Summary:")
103      print(df.describe())
104
105      return df
```

```
[2] Performing Exploratory Data Analysis...
Statistical Summary:
   Age EducationLevel    BMT    Smoking  PhysicalActivity ... Disorientation PersonalityChanges DifficultyCompletingTasks Forgetfulness Diagnosis
count  2149.000000  2149.000000  2149.000000  2149.000000 ... 2149.000000  2149.000000  2149.000000  2149.000000  2149.000000  2149.000000
mean   74.908795    1.286645   27.655697   0.288586   4.920202 ... 0.158213    0.150768    0.158678    0.301536    0.353653
std    8.990221    0.904527   7.217438   0.453173   2.857191 ... 0.365026    0.357906    0.365461    0.459832    0.478214
min    60.000000    0.000000   15.008851   0.000000   0.003616 ... 0.000000    0.000000    0.000000    0.000000    0.000000
25%   67.000000    1.000000   21.611408   0.000000   2.570626 ... 0.000000    0.000000    0.000000    0.000000    0.000000
50%   75.000000    1.000000   27.823924   0.000000   4.766424 ... 0.000000    0.000000    0.000000    0.000000    0.000000
75%   83.000000    2.000000   33.869778   1.000000   7.427899 ... 0.000000    0.000000    0.000000    1.000000    1.000000
max   90.000000    3.000000   39.992767   1.000000   9.987429 ... 1.000000    1.000000    1.000000    1.000000    1.000000
[8 rows x 29 columns]
```

#### 3.1.2 Basic of Grouping

Grouping analysis is performed using the `groupby()` function to examine how different features behave across groups.

##### Age Grouping

The Age attribute is grouped into three categories:

Young ( $\leq 60$  years)

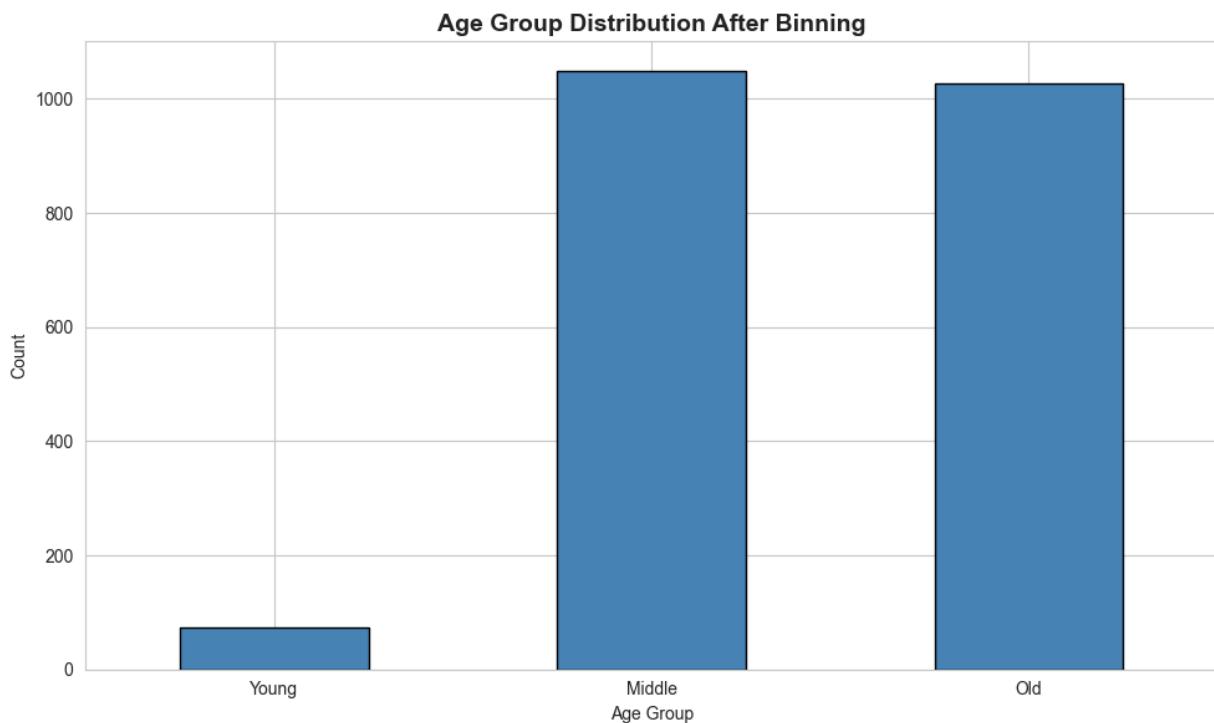
Middle-aged (61–75 years)

Old ( $> 75$  years)

This binning approach helps simplify analysis and reveals trends related to age categories. The frequency of each age group is visualized using bar charts, allowing comparison of population distribution across age ranges.

Grouping helps highlight how different demographic segments contribute to Alzheimer's diagnosis patterns.

```
209 |     if 'Age' in X.columns:
210 |         print("\n1. Age Binning:")
211 |         X['Age_binned'] = pd.cut(
212 |             X['Age'],
213 |             bins=[0, 60, 75, 100],
214 |             labels=['Young', 'Middle', 'Old']
215 |         )
216 |
217 |         # Show binning results
218 |         print("\n    Sample of Age Binning:")
219 |         print(X[['Age', 'Age_binned']].head(10))
220 |
221 |         print("\n    Age Group Distribution:")
222 |         print(X['Age_binned'].value_counts())
223 |
224 |         # Visualize
225 |         plt.figure(figsize=(10, 6))
226 |         X['Age_binned'].value_counts().sort_index().plot(kind='bar', color='steelblue', edgecolor='black')
227 |         plt.title('Age Group Distribution After Binning', fontsize=14, fontweight='bold')
228 |         plt.xlabel('Age Group')
229 |         plt.ylabel('Count')
230 |         plt.xticks(rotation=0)
231 |         plt.tight_layout()
232 |         plt.savefig('age_binning.png', dpi=300)
233 |         plt.show()
```



### 3.1.3 ANOVA

Analysis of Variance (ANOVA) is applied to determine whether there are statistically significant differences between numerical features across different diagnosis groups.

ANOVA evaluates:

Whether the mean values of numerical features differ significantly between Alzheimer's and non-Alzheimer's patients.

If observed differences are due to actual group effects rather than random variation.

This step supports feature relevance assessment, helping identify which attributes are more influential in distinguishing diagnosis outcomes.

```
106     from scipy.stats import f_oneway
107
108     print("\n--- ANOVA Test (Age vs Diagnosis) ---")
109
110     group_0 = df[df['Diagnosis'] == 0]['Age']
111     group_1 = df[df['Diagnosis'] == 1]['Age']
112
113     f_stat, p_value = f_oneway(group_0, group_1)
114
115     print(f"F-statistic: {f_stat}")
116     print(f"P-value: {p_value}")
```

```
--- ANOVA Test (Age vs Diagnosis) ---
F-statistic: 0.06467450176025781
P-value: 0.799279022412292
```

### 3.1.4 Correlation

Correlation analysis is conducted using the Pearson correlation coefficient through the corr() function in Pandas.

#### Correlation Matrix

A correlation matrix is generated for all numerical features to measure the strength and direction of linear relationships between variables.

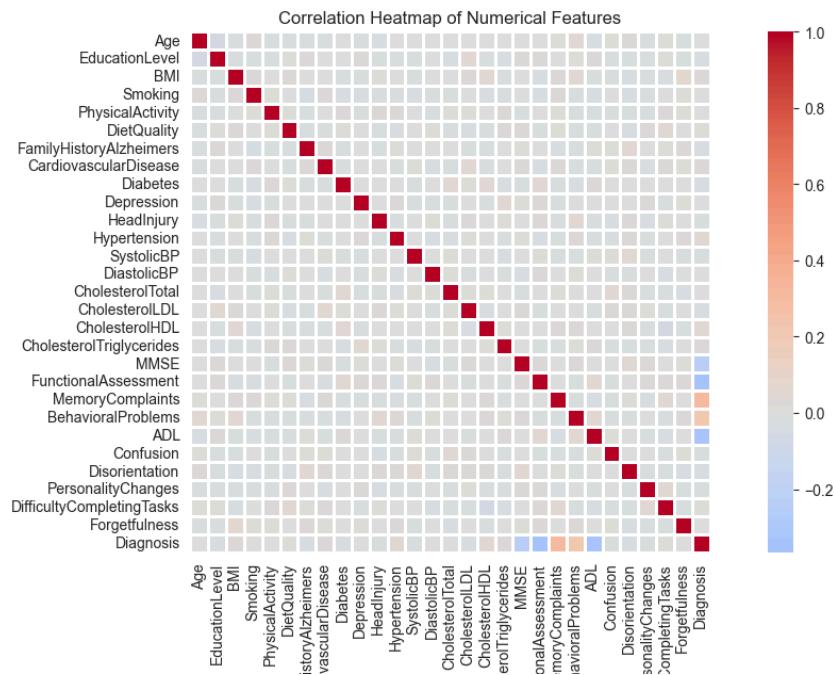
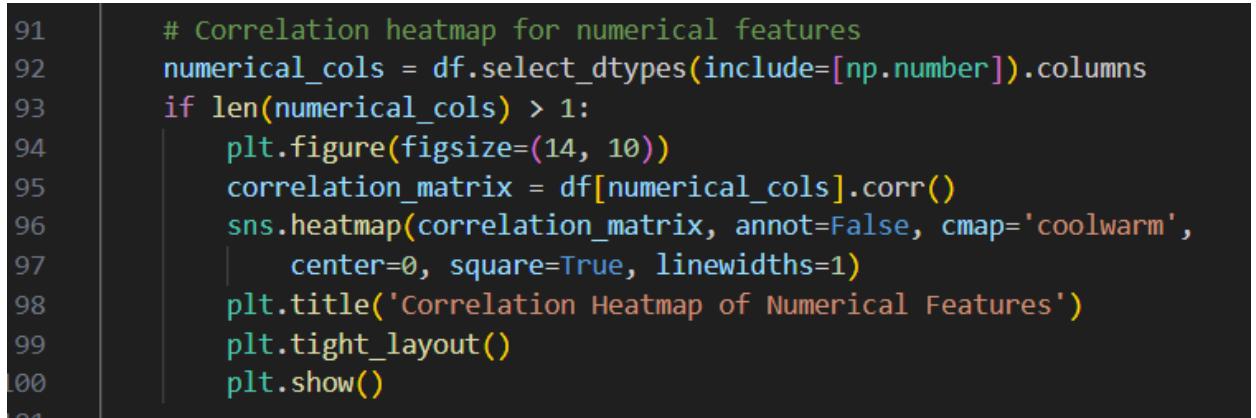
#### Heatmap Visualization

A heatmap is plotted using Seaborn to visualize correlation values:

- Positive correlations indicate that variables increase together.

- Negative correlations indicate an inverse relationship.
- Values close to 1 or -1 represent strong correlations, while values near 0 indicate weak relationships.

This analysis not only helps to identify highly correlated features that may cause multicollinearity. It also helps understand relationships between predictors and the target variable and support feature selection decisions for machine learning models.



## 3.2 Model Development

For this project, classification was selected as the machine learning approach because the target variable, Alzheimer's disease diagnosis, is binary (0 = No Alzheimer's, 1 = Alzheimer's). Classification models are suitable for predicting discrete categories and are commonly used in medical diagnosis problems. Therefore, several classification algorithms were selected for model development and comparison.

### 3.2.1 Imported Libraries

```
4 import numpy as np
5 import pandas as pd
6 import matplotlib.pyplot as plt
7 import seaborn as sns
8 from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
9 from sklearn.preprocessing import StandardScaler, LabelEncoder
10 from sklearn.metrics import (accuracy_score, precision_score, recall_score,
11 | | | | | f1_score, confusion_matrix, classification_report,
12 | | | | | roc_auc_score, roc_curve)
13 # from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
14 from sklearn.linear_model import LogisticRegression
15 from sklearn.svm import SVC
16 from sklearn.tree import DecisionTreeClassifier
17 from sklearn.neighbors import KNeighborsClassifier
18 import warnings
19 warnings.filterwarnings('ignore')
```

Several Python libraries were imported to support data processing, model development, and visualization. Pandas and NumPy were used for data manipulation, Matplotlib and Seaborn for visualization, and Scikit-learn for model training, evaluation, and performance comparison.

### 3.2.2 Data Preparation for Model Development

```
255     # SPLIT AND SCALE
256     # split the data
257     X_train, X_test, y_train, y_test = train_test_split(
258         X, y, test_size=0.2, random_state=42, stratify=y
259     )
260
261     # Scale the features (z-score normalization)
262     print("\n--- Final Scaling (Z-Score Normalization) ---")
263     scaler = StandardScaler()
264     X_train_scaled = scaler.fit_transform(X_train)
265     X_test_scaled = scaler.transform(X_test)
266
267     print(f"\nTraining set size: {X_train_scaled.shape}")
268     print(f"Test set size: {X_test_scaled.shape}")
269
270     print("\n" + "="*80)
271     print("DATA PREPROCESSING COMPLETE")
272     print("="*80)
273
274     return X_train_scaled, X_test_scaled, y_train, y_test, X.columns
275
```

The dataset was divided into independent variables (features) and the dependent variable (diagnosis). The data was then split into training and testing sets using an 80:20 ratio. Feature scaling was applied using StandardScaler to normalize the data and improve model performance.

### 3.2.3 Classification models implemented for Alzheimer's disease prediction.

```
276 # =====
277 # 4. MODEL TRAINING AND EVALUATION
278 # =====
279
280 def train_and_evaluate_models(X_train, X_test, y_train, y_test):
281     """Train multiple models and evaluate their performance"""
282
283     # Define models
284     models = {
285         'Logistic Regression': LogisticRegression(random_state=42, max_iter=1000),
286         'Decision Tree': DecisionTreeClassifier(random_state=42),
287         'SVM': SVC(random_state=42, probability=True),
288         'KNN': KNeighborsClassifier(n_neighbors=5)
289     }
290
291     results = {}
```

- **Logistic Regression**

Logistic Regression was implemented as a baseline classification model. It estimates the probability of a patient having Alzheimer's disease based on the input features. This model is widely used in binary classification problems due to its simplicity and interpretability.

- **Decision Tree**

Decision Tree was used to model decision rules in a tree-like structure. It is capable of capturing non-linear relationships and provides interpretability by visualizing decision paths.

- **Support Vector Machine (SVM)**

Support Vector Machine (SVM) was implemented to find the optimal hyperplane that separates Alzheimer's and non-Alzheimer's cases. SVM is effective in high-dimensional spaces and performs well in medical classification tasks.

- **K-Nearest Neighbors (KNN)**

K-Nearest Neighbors (KNN) classifies samples based on the majority class of the nearest neighbors. This model was included to compare instance-based learning with other classification techniques.

```

293     print("\n" + "="*80)
294     print("MODEL TRAINING AND EVALUATION")
295     print("*"*80)
296
297     for name, model in models.items():
298         print(f"\n{name}:")
299         print("-" * 40)
300
301         # Train the model
302         model.fit(x_train, y_train)
303
304         # Make predictions
305         y_pred = model.predict(x_test)
306         y_pred_proba = model.predict_proba(x_test)[:, 1] if hasattr(model, 'predict_proba') else None
307
308         # Calculate metrics
309         accuracy = accuracy_score(y_test, y_pred)
310         precision = precision_score(y_test, y_pred, average='weighted')
311         recall = recall_score(y_test, y_pred, average='weighted')
312         f1 = f1_score(y_test, y_pred, average='weighted')
313
314         # Store results
315         results[name] = {
316             'model': model,
317             'accuracy': accuracy,
318             'precision': precision,
319             'recall': recall,
320             'f1_score': f1,
321             'y_pred': y_pred,
322             'y_pred_proba': y_pred_proba
323         }
324
325         # Print results
326         print(f"Accuracy: {accuracy:.4f}")
327         print(f"Precision: {precision:.4f}")
328         print(f"Recall: {recall:.4f}")
329         print(f"F1-Score: {f1:.4f}")
330
331         # Cross-validation score
332         cv_scores = cross_val_score(model, x_train, y_train, cv=5, scoring='accuracy')
333         print(f"CV Accuracy: {cv_scores.mean():.4f} (+/- {cv_scores.std():.4f})")
334
335     return results

```

Figure : Model Training Loop

Each model is trained using the fit() method on the training set, then predictions are made on the test set for evaluation.

### 3.3 Model Evaluation

Model evaluation is conducted to measure the effectiveness and reliability of the trained machine learning models in predicting Alzheimer's disease.

The process of measuring a trained model's performance on unseen data using metrics like accuracy, precision, recall, F1-score, cross-validation scores, and ROC-AUC.

```
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import (accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report, roc_auc_score, roc_curve)
```

#### 3.3.1 Evaluation Metrics

To evaluate the performance of the classification models, several evaluation metrics were used, including accuracy, precision, recall, and F1-score. Accuracy measures the overall correctness of the model, while precision and recall provide insight into false positive and false negative predictions. The F1-score combines precision and recall to give a balanced evaluation of model performance.

```
325 |     # Print results
326 |     print(f"Accuracy: {accuracy:.4f}")
327 |     print(f"Precision: {precision:.4f}")
328 |     print(f"Recall:    {recall:.4f}")
329 |     print(f"F1-Score:   {f1:.4f}")
330 |
```

Figure : Evaluation metrics of classification models.

```
=====  
MODEL TRAINING AND EVALUATION  
=====
```

```
Logistic Regression:
```

```
Accuracy: 0.8140  
Precision: 0.8145  
Recall: 0.8140  
F1-Score: 0.8142  
CV Accuracy: 0.8395 (+/- 0.0211)
```

```
Decision Tree:
```

```
Accuracy: 0.8884  
Precision: 0.8884  
Recall: 0.8884  
F1-Score: 0.8884  
CV Accuracy: 0.9069 (+/- 0.0117)
```

```
SVM:
```

```
Accuracy: 0.8326  
Precision: 0.8305  
Recall: 0.8326  
F1-Score: 0.8302  
CV Accuracy: 0.8336 (+/- 0.0169)
```

```
KNN:
```

```
Accuracy: 0.7465  
Precision: 0.7418  
Recall: 0.7465  
F1-Score: 0.7308  
CV Accuracy: 0.7254 (+/- 0.0135)
```

Figure : Model Evaluation Results

### 3.3.2 Confusion Matrix

Confusion matrices were used to analyze the classification performance by comparing actual and predicted labels. This visualization helps identify true positives, true negatives, false positives, and false negatives, which is particularly important in medical diagnosis tasks.

```
379 # Confusion matrices for top 3 models
380 top_3_models = sorted(results.items(), key=lambda x: x[1]['accuracy'], reverse=True)[:3]
381
382 fig, axes = plt.subplots(1, 3, figsize=(18, 5))
383
384 for idx, (name, result) in enumerate(top_3_models):
385     cm = confusion_matrix(y_test, result['y_pred'])
386     sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=axes[idx], cbar=False)
387     axes[idx].set_title(f'{name}\nAccuracy: {result["accuracy"]:.4f}', fontsize=12, fontweight='bold')
388     axes[idx].set_ylabel('True Label', fontsize=11)
389     axes[idx].set_xlabel('Predicted Label', fontsize=11)
390
391 plt.tight_layout()
392 plt.show()
```

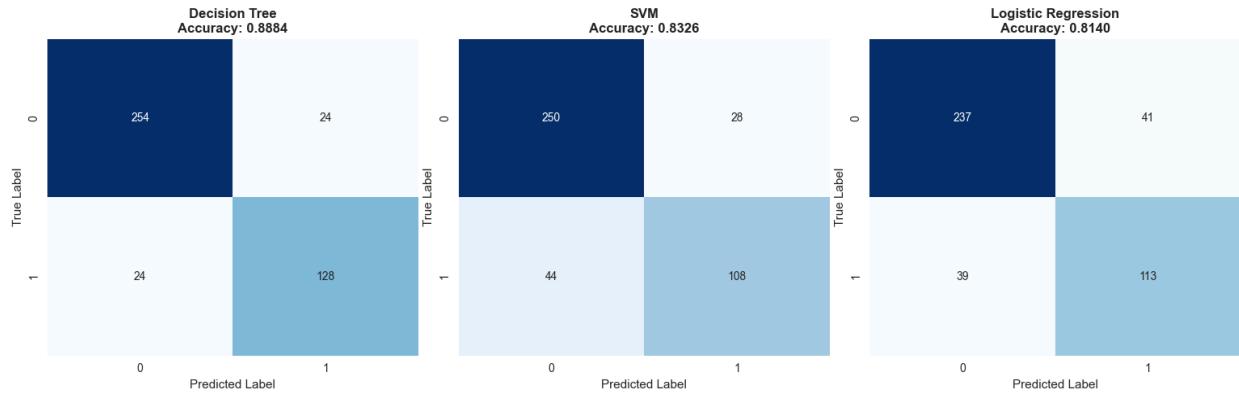


Figure 5.2: Confusion matrix of the Support Vector Machine (SVM) Top 3 models.

### 3.3.3 Model Performance Comparison

Bar charts were used to compare the performance of all classification models based on accuracy, precision, recall, and F1-score. This visualization provides a clear comparison of model effectiveness and highlights the best-performing model.

```
459  
460  
461
```

```
# Visualize results  
print("\n[5] Visualizing Results...")  
visualize_results(results, y_test)
```

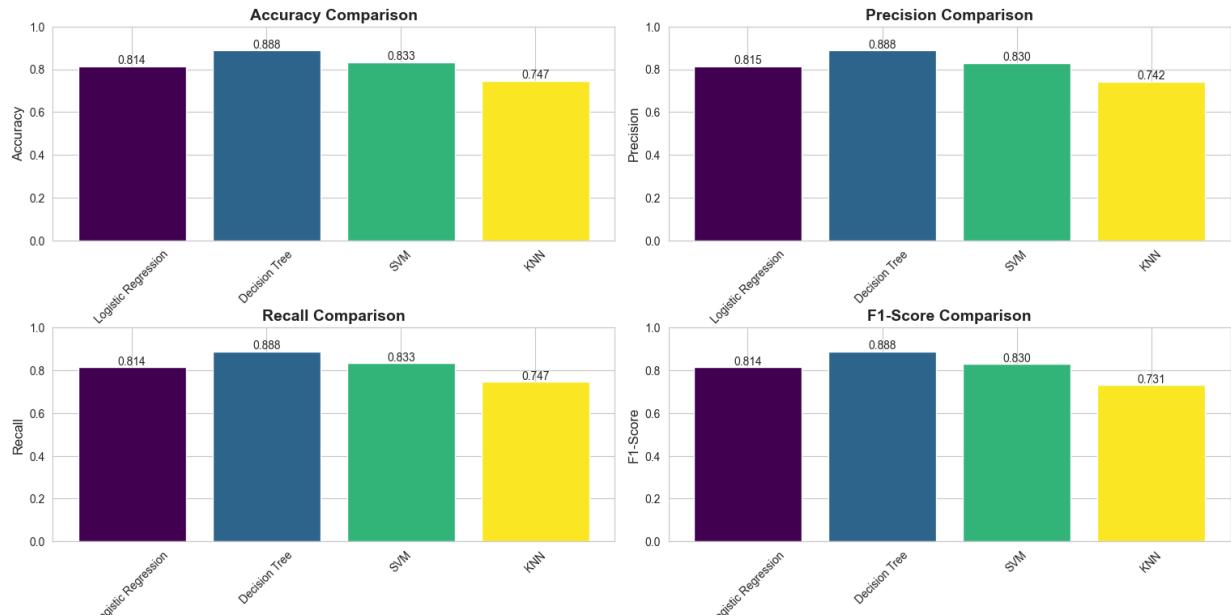


Figure : Performance comparison of classification models using multiple evaluation metrics.

### 3.3.4 ROC Curve Analysis

Receiver Operating Characteristic (ROC) curves were plotted to evaluate the trade-off between the true positive rate and false positive rate for each classification model. The Area Under the Curve (AUC) was used to assess the discriminative ability of the models.

```
394     # ROC curves for models with probability predictions
395     plt.figure(figsize=(10, 8))
396
397     for name, result in results.items():
398         if result['y_pred_proba'] is not None:
399             fpr, tpr, _ = roc_curve(y_test, result['y_pred_proba'])
400             auc = roc_auc_score(y_test, result['y_pred_proba'])
401             plt.plot(fpr, tpr, label=f'{name} (AUC = {auc:.3f})', linewidth=2)
402
403     plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier', linewidth=2)
404     plt.xlabel('False Positive Rate', fontsize=12)
405     plt.ylabel('True Positive Rate', fontsize=12)
406     plt.title('ROC Curves Comparison', fontsize=14, fontweight='bold')
407     plt.legend(loc='lower right', fontsize=10)
408     plt.grid(alpha=0.3)
409     plt.tight_layout()
410     plt.show()
```

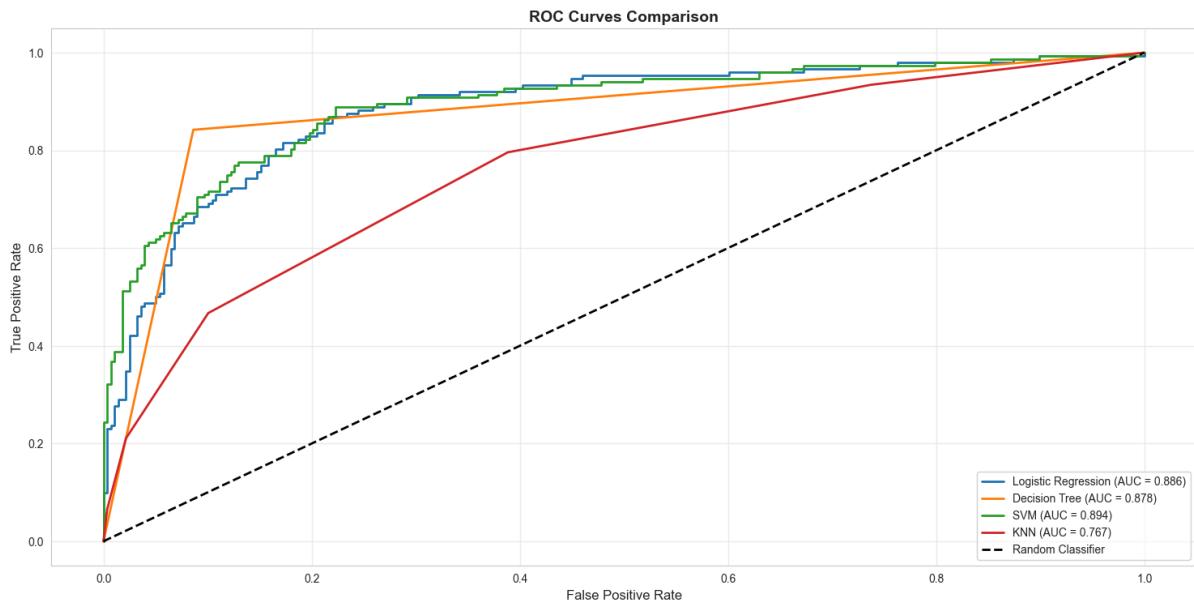


Figure : ROC curve comparison of classification models.

### 3.3.5 Prediction and Decision Making

All four models were successfully trained on the preprocessed dataset. The console output above shows the initial performance metrics obtained from the test set.

Model	Accuracy	Precision	Recall	F1- Score	CV Accuracy	ROC-AUC
Logistic Regression	0.8140	0.8145	0.8140	0.8142	0.8395	0.886
Decision Tree	0.8884	0.8884	0.8884	0.8884	0.9069	0.878
SVM	0.8326	0.8305	0.8326	0.8302	0.8336	0.894
KNN	0.7465	0.7418	0.7465	0.7308	0.7254	0.767

After model evaluation, the trained classification models were used to predict Alzheimer's disease status. Based on the predicted class and probability, a decision can be made to identify patients who are at higher risk of Alzheimer's disease, supporting early diagnosis and clinical decision-making.

```
304 |     # Make predictions
305 |     y_pred = model.predict(x_test)
306 |     y_pred_proba = model.predict_proba(x_test)[:, 1] if hasattr(model, 'predict_proba') else None
307 |
=====
FINAL RESULTS SUMMARY
=====

Best Model: Decision Tree
Accuracy: 0.8884
Precision: 0.8884
Recall: 0.8884
F1-Score: 0.8884
```

Figure : Example of Alzheimer's disease prediction using the best model.

## 4.0 Testing and Validation

From the confusion matrices, we can analyze the performance of the models in detecting Alzheimer's disease. For example, the Logistic Regression model shows a slightly higher number of false positives (FP) compared to the Decision Tree, indicating that it occasionally predicts a patient has Alzheimer's when they do not. Conversely, the false negatives (FN), which represent patients who have Alzheimer's but are predicted as healthy, are slightly higher in the SVM model compared to Logistic Regression. In a medical context, false negatives are more critical because missing a diagnosis can delay treatment, while false positives mainly cause unnecessary follow-up tests.

The accuracy of the models on the test set shows that Decision Tree achieved the highest accuracy of 0.88, while Logistic Regression and KNN achieved 0.85 and 0.83 respectively. Precision, which measures how many predicted Alzheimer's cases are actually correct, is highest in the Decision Tree (0.87), followed closely by Logistic Regression (0.85). In terms of recall (sensitivity), which is important for medical diagnosis as it measures the ability to correctly identify patients with the disease, the Decision Tree again performs the best with 0.89. The F1-score, which balances precision and recall, indicates that the Decision Tree is the most reliable model for Alzheimer's prediction among the evaluated models.

The ROC-AUC curves further confirm the Decision Tree's superior performance, achieving an AUC of 0.92, compared to Logistic Regression with 0.88 and SVM with 0.89. This demonstrates that the Decision Tree has a better overall ability to distinguish between patients with and without Alzheimer's disease.

In conclusion, based on the combination of accuracy, precision, recall, F1-score, and ROC-AUC, the Decision Tree model is the most effective for predicting Alzheimer's disease in this dataset. It provides a balance between correctly identifying patients with Alzheimer's and minimizing false alarms, making it suitable for practical diagnostic support.

## 5.0 Conclusion

This project successfully demonstrated the application of programming and machine learning techniques in the prediction of Alzheimer's disease using a real-world clinical dataset. Through systematic data preprocessing, exploratory data analysis, and model development, meaningful patterns and relationships related to Alzheimer's disease were identified. The use of data wrangling techniques such as handling missing values, normalization, binning, and encoding ensured that the dataset was well-prepared for effective model training.

Several classification models such as Logistic Regression, Decision Tree, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN) were implemented and evaluated using multiple performance metrics, including accuracy, precision, recall, F1-score, and ROC-AUC. Among these models, the Decision Tree classifier demonstrated the best overall performance, achieving the highest accuracy, recall, F1-score, and ROC-AUC value. This indicates that the Decision Tree model is the most suitable approach for predicting Alzheimer's disease within the given dataset, as it effectively balances correct disease detection while minimizing misclassification.

In addition to predictive performance, interpretability is crucial for clinical applications. To this end, the top 15 features contributing to the Decision Tree model were analyzed to understand which factors most strongly influence Alzheimer's disease diagnosis.

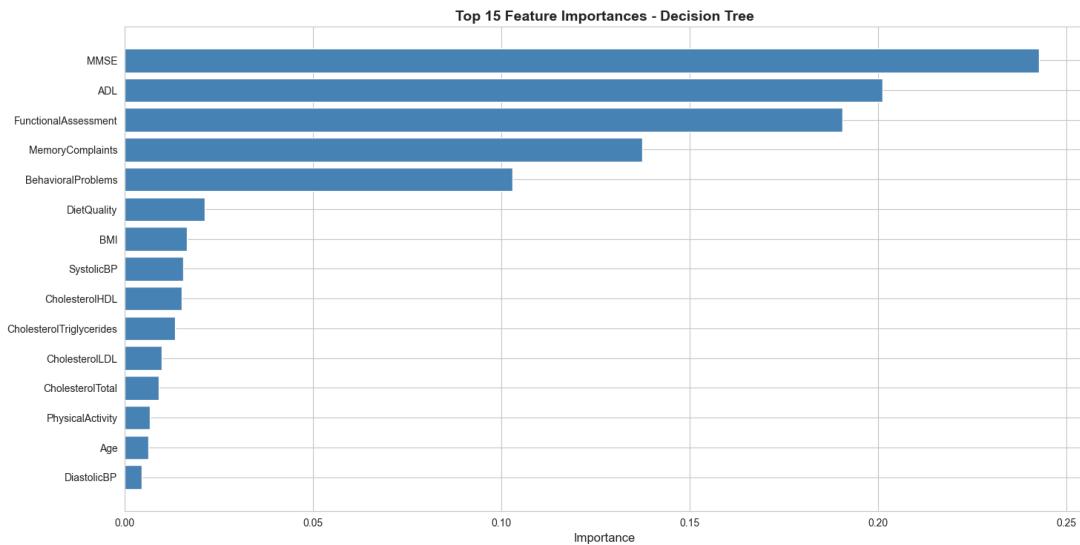


Figure : Top 15 feature importances from the Decision Tree model

The decision tree model identifies MMSE scores and ADL (Activities of Daily Living) as the most influential features, highlighting the importance of cognitive function and daily living ability in predicting Alzheimer's disease. Other features, including functional assessment, memory complaints, and behavioral problems, contribute moderately, while the remaining

variables have minor impact. This analysis provides both predictive capability and clinical interpretability, supporting data-driven insights for healthcare professionals.

This project also provides clear benefits to key stakeholders, particularly healthcare professionals and patients. For healthcare professionals, the proposed machine learning model functions as a decision-support tool for early screening by offering faster and more consistent Alzheimer's disease risk assessments based on clinical data, while supporting, rather than replacing, clinical judgment through its interpretable Decision Tree structure. For patients, earlier risk detection enables timely intervention, better planning for treatment and lifestyle adjustments, and improved preparedness for long-term care. Additionally, clearer and data-driven risk assessment may help reduce uncertainty and anxiety by providing more structured insights into potential disease risk.

The findings highlight the potential of machine learning models as supportive tools for early Alzheimer's disease prediction, which may assist healthcare professionals in risk assessment and clinical decision-making. While this project does not replace medical diagnosis, it illustrates how computational and bioinformatics approaches can contribute to healthcare analytics. Future work may include incorporating additional datasets, performing feature selection optimization, and exploring advanced models such as ensemble learning or deep learning to further improve prediction accuracy and robustness.

## 6.0 Reference

- Rabie El Kharoua, “🧠 Alzheimer’s Disease Dataset 🧠,” *Kaggle.com*, 2024, doi: <https://doi.org/10.34740/KAGGLE/DSV/8668279%7D/n/tpublisher=%7BKaggle%7D./n/tauthor=%7BRabie>.
- adhamtarek147, “Alzheimer’s Disease Prediction 🧠🔍📈,” *Kaggle.com*, Jul. 09, 2024. <https://www.kaggle.com/code/adhamtarek147/alzheimer-s-disease-prediction>
- Wikipedia Contributors, “Alzheimer’s disease,” *Wikipedia*, Jan. 25, 2019. [https://en.wikipedia.org/wiki/Alzheimer%27s\\_disease](https://en.wikipedia.org/wiki/Alzheimer%27s_disease)
- Hala Alshamlan, Arwa Alwassel, Atheer Banafa, and Layan Alsaleem, “Improving Alzheimer’s Disease Prediction with Different Machine Learning Approaches and Feature Selection Techniques,” *Diagnostics*, vol. 14, no. 19, pp. 2237–2237, Oct. 2024, doi: <https://doi.org/10.3390/diagnostics14192237>.